Use Reducer

Ex4:

Code:

import React, { useReducer, useState } from "react";

import { Button, Form, ListGroup, InputGroup } from "react-bootstrap";

function listReducer(state, action) {

  switch (action.type) {

    case "ADD\_ITEM":

      return { ...state, items: [...state.items, action.item] };

    case "REMOVE\_ITEM":

      return { ...state, items: state.items.filter(item => item.id !== action.id) };

    case "EDIT\_ITEM":

      return {

        ...state,

        items: state.items.map(item => item.id === action.id ? { ...item, name: action.name } : item)

      };

    case "SORT\_ALPHA":

      return { ...state, items: [...state.items].sort((a, b) => a.name.localeCompare(b.name)) };

    case "SORT\_TIME":

      return { ...state, items: [...state.items].sort((a, b) => a.id - b.id) };

    default:

      return state;

  }

}

export default function ItemListAdvanced() {

  const [state, dispatch] = useReducer(listReducer, { items: [] });

  const [newItemName, setNewItemName] = useState("");

  const [filter, setFilter] = useState("");

  const [editingId, setEditingId] = useState(null);

  const [editingValue, setEditingValue] = useState("");

  const handleAdd = () => {

    if (!newItemName.trim()) return;

    dispatch({ type: "ADD\_ITEM", item: { id: Date.now(), name: newItemName } });

    setNewItemName("");

  };

  const handleEdit = (id, name) => {

    setEditingId(id);

    setEditingValue(name);

  };

  const handleSaveEdit = (id) => {

    dispatch({ type: "EDIT\_ITEM", id, name: editingValue });

    setEditingId(null);

    setEditingValue("");

  };

  const filteredItems = state.items.filter(item => item.name.toLowerCase().includes(filter.toLowerCase()));

  return (

    <div>

      <InputGroup className="mb-3">

        <Form.Control placeholder="Search..." value={filter} onChange={e => setFilter(e.target.value)} />

      </InputGroup>

      <InputGroup className="mb-3">

        <Form.Control value={newItemName} onChange={e => setNewItemName(e.target.value)} placeholder="Enter item" />

        <Button onClick={handleAdd}>Add</Button>

      </InputGroup>

      <div className="mb-2">

        <Button onClick={() => dispatch({ type: "SORT\_ALPHA" })}>Sort A-Z</Button>{' '}

        <Button onClick={() => dispatch({ type: "SORT\_TIME" })}>Sort by Time</Button>

      </div>

      <ListGroup>

        {filteredItems.map(item => (

          <ListGroup.Item key={item.id} className="d-flex justify-content-between align-items-center">

            {editingId === item.id ? (

              <InputGroup>

                <Form.Control

                  value={editingValue}

                  onChange={(e) => setEditingValue(e.target.value)}

                />

                <Button variant="success" onClick={() => handleSaveEdit(item.id)}>Save</Button>

              </InputGroup>

            ) : (

              <>

                <span>{item.name}</span>

                <div>

                  <Button variant="outline-primary" className="me-2" onClick={() => handleEdit(item.id, item.name)}>

                    Edit

                  </Button>

                  <Button variant="outline-danger" onClick={() => dispatch({ type: "REMOVE\_ITEM", id: item.id })}>

                    Remove

                  </Button>

                </div>

              </>

            )}

          </ListGroup.Item>

        ))}

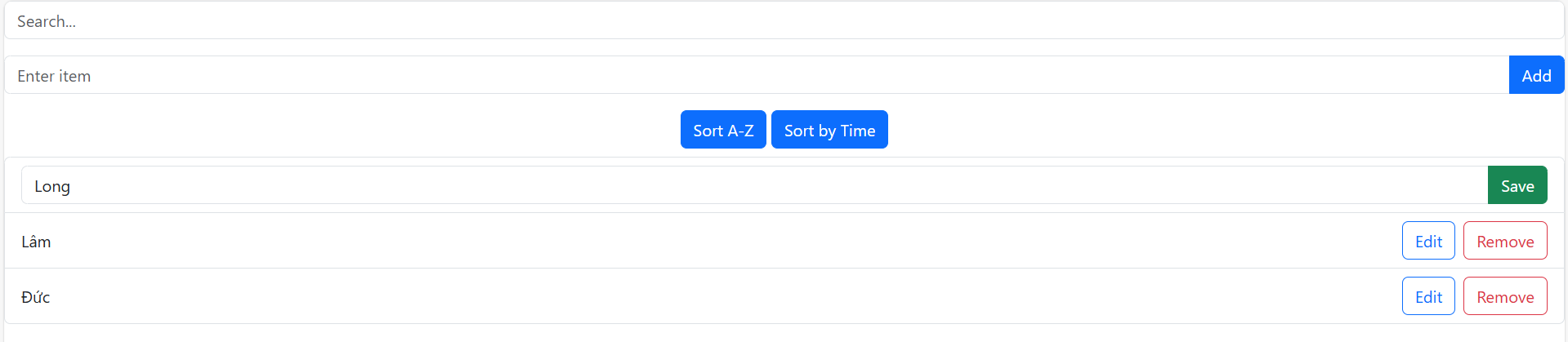
      </ListGroup>

    </div>

  );

}

Kết Quả:



Ex5:

Code:

import React, { useReducer } from "react";

import { Button, Container, Card } from "react-bootstrap";

const initialState = {

  questions: [

    {

      id: 1,

      question: "What is the capital of Australia?",

      options: ["Sydney", "Canberra", "Melbourne", "Perth"],

      answer: "Canberra"

    },

    {

      id: 2,

      question: "Which planet is known as the Red Planet?",

      options: ["Venus", "Mars", "Jupiter", "Saturn"],

      answer: "Mars"

    },

    {

      id: 3,

      question: "What is the largest ocean on Earth?",

      options: ["Atlantic Ocean", "Indian Ocean", "Pacific Ocean", "Arctic Ocean"],

      answer: "Pacific Ocean"

    }

  ],

  currentQuestion: 0,

  selectedOption: "",

  score: 0,

  showScore: false

};

function quizReducer(state, action) {

  switch (action.type) {

    case "SELECT\_OPTION":

      return { ...state, selectedOption: action.payload };

    case "NEXT\_QUESTION":

      const isCorrect = state.selectedOption === state.questions[state.currentQuestion].answer;

      return {

        ...state,

        score: isCorrect ? state.score + 1 : state.score,

        currentQuestion: state.currentQuestion + 1,

        selectedOption: "",

        showScore: state.currentQuestion + 1 === state.questions.length

      };

    case "RESTART\_QUIZ":

      return { ...initialState };

    default:

      return state;

  }

}

export default function QuestionBank() {

  const [state, dispatch] = useReducer(quizReducer, initialState);

  const { questions, currentQuestion, selectedOption, score, showScore } = state;

  const handleOptionSelect = (option) => {

    dispatch({ type: "SELECT\_OPTION", payload: option });

  };

  const handleNextQuestion = () => {

    dispatch({ type: "NEXT\_QUESTION" });

  };

  const handleRestartQuiz = () => {

    dispatch({ type: "RESTART\_QUIZ" });

  };

  return (

    <Container className="mt-4">

      <Card className="p-4">

        {showScore ? (

          <div className="text-center">

            <h2>Your Score: {score} / {questions.length}</h2>

            <Button variant="primary" onClick={handleRestartQuiz}>

              Restart Quiz

            </Button>

          </div>

        ) : (

          <div>

            <h4>Question {questions[currentQuestion].id}:<br />{questions[currentQuestion].question}</h4>

            <div className="mt-3">

              {questions[currentQuestion].options.map((option, index) => (

                <Button

                  key={index}

                  variant={selectedOption === option ? "success" : "outline-secondary"}

                  className="m-2"

                  onClick={() => handleOptionSelect(option)}

                >

                  {option}

                </Button>

              ))}

            </div>

            <Button

              variant="primary"

              className="mt-3"

              disabled={!selectedOption}

              onClick={handleNextQuestion}

            >

              {currentQuestion === questions.length - 1 ? "Finish Quiz" : "Next Question"}

            </Button>

          </div>

        )}

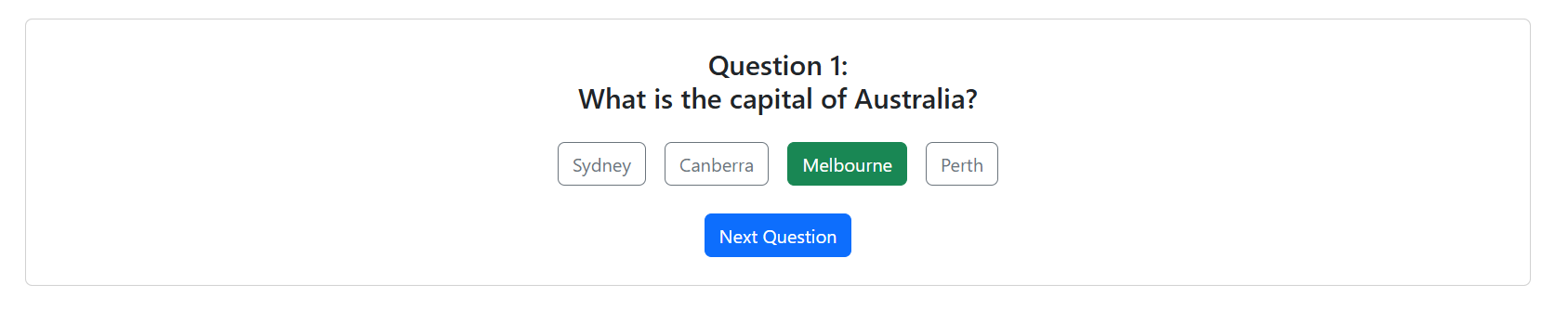
      </Card>

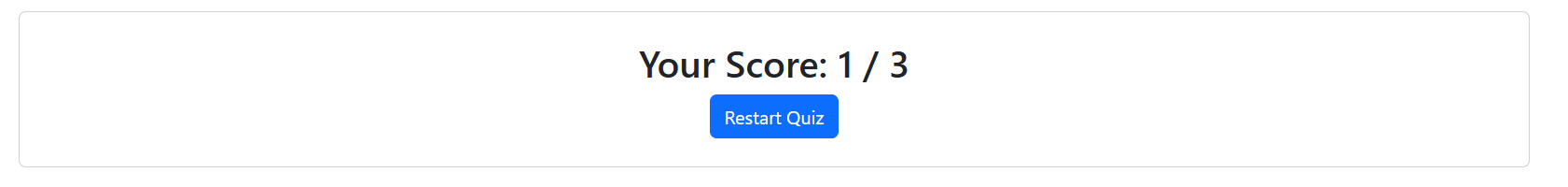
    </Container>

  );

}

Kết Quả:





Ex6:

Code:

import React, { useEffect, useReducer } from "react";

import { FaCheckCircle, FaTimesCircle } from "react-icons/fa";

const quizInitial = {

  questions: [

    {

      id: 1,

      question: "Capital of Australia?",

      options: ["Sydney", "Canberra", "Melbourne", "Perth"],

      answer: "Canberra"

    },

    {

      id: 2,

      question: "Red Planet?",

      options: ["Venus", "Mars", "Jupiter", "Saturn"],

      answer: "Mars"

    },

    {

      id: 3,

      question: "Largest ocean?",

      options: ["Atlantic", "Indian", "Pacific", "Arctic"],

      answer: "Pacific"

    }

  ],

  currentQuestion: 0,

  selectedOption: "",

  score: 0,

  showScore: false,

  feedback: "",

  timer: 10,

  highScore: localStorage.getItem("highScore") || 0

};

function quizReducer(state, action) {

  switch (action.type) {

    case "SELECT\_OPTION":

      const selected = action.payload;

      const correct = state.questions[state.currentQuestion].answer;

      return {

        ...state,

        selectedOption: selected,

        feedback: selected === correct ? "correct" : "incorrect"

      };

    case "NEXT\_QUESTION":

      const correctAns = state.selectedOption === state.questions[state.currentQuestion].answer;

      const newScore = correctAns ? state.score + 1 : state.score;

      const finished = state.currentQuestion + 1 === state.questions.length;

      const newHigh = Math.max(newScore, state.highScore);

      if (finished) localStorage.setItem("highScore", newHigh);

      return {

        ...state,

        score: newScore,

        currentQuestion: state.currentQuestion + 1,

        selectedOption: "",

        feedback: "",

        showScore: finished,

        timer: 10,

        highScore: newHigh

      };

    case "TICK":

      return { ...state, timer: state.timer - 1 };

    case "RESTART":

      return { ...quizInitial, highScore: state.highScore };

    default:

      return state;

  }

}

export default function QuestionBankEnhanced() {

  const [state, dispatch] = useReducer(quizReducer, quizInitial);

  const { questions, currentQuestion, selectedOption, score, showScore, feedback, timer, highScore } = state;

  useEffect(() => {

    if (showScore) return;

    if (timer === 0) {

      dispatch({ type: "NEXT\_QUESTION" });

    }

    const interval = setInterval(() => dispatch({ type: "TICK" }), 1000);

    return () => clearInterval(interval);

  }, [timer, showScore]);

  return (

    <div>

      {showScore ? (

        <div>

          <h2>Score: {score}/{questions.length}</h2>

          <h3>High Score: {highScore}</h3>

          <button onClick={() => dispatch({ type: "RESTART" })}>Restart</button>

        </div>

      ) : (

        <div>

          <h4>Question {currentQuestion + 1}/{questions.length}</h4>

          <p>{questions[currentQuestion].question}</p>

          <p style={{ color: timer < 5 ? 'red' : 'black' }}>⏱️ {timer}s</p>

          {questions[currentQuestion].options.map(option => (

            <button key={option} onClick={() => dispatch({ type: "SELECT\_OPTION", payload: option })}>

              {option}

            </button>

          ))}

          {feedback === "correct" && <p><FaCheckCircle color="green" /> Correct! 🎉</p>}

          {feedback === "incorrect" && <p><FaTimesCircle color="red" /> Incorrect! Answer: {questions[currentQuestion].answer}</p>}

          <button onClick={() => dispatch({ type: "NEXT\_QUESTION" })} disabled={!selectedOption}>

            {currentQuestion === questions.length - 1 ? "Finish Quiz" : "Next Question"}

          </button>

        </div>

      )}

    </div>

  );

}

Kết Quả:

